

Biodata

Nama : Ari Cahyono
No-Reg : 149368582101-638
Kelas : DTS Scalable Web Service With GOLANG

Panduan Pembuatan Program:

1. Buat Folder Mygram
2. buka terminal dan jalankan go mod init Mygram
3. buat file main.go
4. instal beberapa package
 - GORM : untuk ORM
 - GIN : untuk Routing
 - dto-mapper : untuk membantu convert model ke DTO
 - govalidator : untuk membantu dalam validasi data
 - jwt-go : untuk membantu dalam membuat token JWT
5. Buat beberapa folder
 - controllers
 - models
 - routers
 - middlewares
 - configs
 - dto
 - helpers
 - temp
6. buat model
 - user
 - photo
 - comment
 - social media
7. buat file pada folder config dengan nama config db.go dan daftarkan model yang digunakan
8. buat beberapa responseDtO pada folder dto
9. buat beberapa helper pada folder helper
10. buat controller pada folder controller, kemudian gunakan dto dan beberapa helpers yang sudah di buat pada setiap func di controller.
 - userControllers.go (CRUD)
 - photoControllers.go (CRUD)
 - commentControllers.go (CRUD)
 - socialmediaControllers.go (CRUD)
11. buat file authentication.go pada folder middlewares
12. buat file pada folder router dengan nama router.go buat endpoint api, hubungkan setiap endpoint dengan controller yang digunakan dan batasi beberapa endpoint dengan middlewares yang di butuhkan
13. Selesai.

Panduan Menjalankan Program:

BE:

1. Siapkan Database Postgres,
kemudian buat Database dengan nama
- mygram
2. jalankan program dengan perintah
- go run main.go

Referensi / Package yang digunakan:

- GORM : untuk ORM
- GIN : untuk Routing
- dto-mapper : untuk membantu convert model ke DTO
- - govalidator : untuk membantu dalam validasi data
- - jwt-go : untuk membantu dalam membuat token JWT

Postman :

URL Collection API :

- <https://www.getpostman.com/collections/5712177cc898d4fe2925>

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/users/login`. The request body is a JSON object with `email` and `password` fields. The response is a 200 OK status with a 32 ms response time and a 261 B body. The response body is a JSON object containing a `token` field.

```
{
  "email": "test@gmail.com",
  "password": "test123"
}
```

200 OK 32 ms 261 B

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1bWVpbCI6InRlc3RAZ21haWwuc29tIiwiaWF0IjF9LmFpLm5mLmCuOktlZppoh63XvhbbF01GD0SLm5UREoGg"
}
```

Update User

PUT : <http://localhost:3000/users?userId=1>

Use Authorization : Bearer Token

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/users?userId=1`. The request body is a JSON object with the following data:

| KEY | VALUE | DESCRIPTION |
|-------------------|-----------------------------|-------------|
| username | test_updated | |
| email | test_updated@gmail.com | |
| profile_image_url | YwELK3eRM/dream-is-real.jpg | |

The response is a JSON object with a status of 200 OK, 28 ms, and 275 B. The response body is shown in the 'Body' tab, displaying the following JSON:

```
{  "data": {    "user_id": 1,    "username": "test_updated",    "email": "test_updated@gmail.com",    "age": 0,    "updated_at": "2022-10-21T22:14:39.4312867+07:00"  },  "status": 200}
```

Delete User

DELETE : <http://localhost:3000/users>

Use Authorization : Bearer Token

The screenshot shows a REST client interface with a DELETE request to `http://localhost:3000/users`. The request is configured with the following settings:

- Type: Bearer...
- Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

The response is a JSON object with a status of 200 OK, 15 ms, and 192 B. The response body is shown in the 'Body' tab, displaying the following JSON:

```
{  "message": "Your account has been successfully deleted",  "status": 200}
```

PhotoController

Post Photo

POST : <http://localhost:3000/photos>

Use Authorization : Bearer Token

POST <http://localhost:3000/photos> Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

| | KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-------------------------------------|---------|-----------------------------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> | photo | KHslGoeZ4/dream-is-real.jpg | | | |
| <input checked="" type="checkbox"/> | title | test231 | | | |
| <input checked="" type="checkbox"/> | caption | hello | | | |

Body Cookies Headers (3) Test Results 201 Created 25 ms 310 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "photo_id": 1,
4     "title": "test231",
5     "caption": "hello",
6     "photo_url": "fHK5a84jjJkwzDkh9-1666366042.jpg",
7     "user_url": 1,
8     "created_at": "2022-10-21T22:27:22.8007736+07:00"
9   },
10  "status": 201
11 }
```

Get All Photo

GET : <http://localhost:3000/photos>

Use Authorization : Bearer Token

GET <http://localhost:3000/photos> Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (3) Test Results 200 OK 10 ms 665 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": [
3     {
4       "photo_id": 1,
5       "title": "test231",
6       "caption": "hello",
7       "photo_url": "fHK5a84jjJkwzDkh9-1666366042.jpg",
8       "user_url": 1,
9       "created_at": "2022-10-21T22:27:22.8007736+07:00",
10      "updated_at": "2022-10-21T22:27:22.8007736+07:00",
11      "User": {
12        "username": "test",
13        "email": "test@gmail.com"
14      }
15    }
16  ]
17 }
```

Update Photo

PUT : <http://localhost:3000/photos/1>

Use Authorization : Bearer Token

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/photos/1`. The request body is a JSON object with the following data:

| KEY | VALUE | DESCRIPTION |
|---------|-----------------------------|-------------|
| photo | j-BOGz4Em/dream-is-real.jpg | |
| title | 123 | |
| caption | 123 | |

The response is a JSON object with a status of 200:

```
{
  "data": {
    "photo_id": 1,
    "title": "123",
    "caption": "123",
    "photo_url": "V3vC5AWX39IVUWSP2-1666366135.jpg",
    "user_url": 1,
    "updated_at": "2022-10-21T22:28:55.7688622+07:00"
  },
  "status": 200
}
```

Delete Photo

Delete : <http://localhost:3000/photos/1>

The screenshot shows a REST client interface with a DELETE request to `http://localhost:3000/photos/1`. The request is configured with Bearer Token authorization. The response is a JSON object with a status of 200:

```
{
  "message": "Your photo has been successfully deleted",
  "status": 200
}
```

CommentController

Post Comment

POST : <http://localhost:3000/comments>

Use Authorization : Bearer Token

POST <http://localhost:3000/comments> Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {
2   "message": "keren bro",
3   "photo_id": 1
4 }
```

Body Cookies Headers (3) Test Results 201 Created 22 ms 263 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "comment_id": 1,
4     "message": "keren bro",
5     "user_id": 1,
6     "photo_id": 1,
7     "created_at": "2022-10-21T22:36:04.9621303+07:00"
8   },
9   "status": 201
10 }
```

Get All Comments

GET : <http://localhost:3000/comments>

Use Authorization : Bearer Token

GET <http://localhost:3000/comments> Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Body Cookies Headers (3) Test Results 200 OK 14 ms 482 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": [
3     {
4       "comment_id": 1,
5       "message": "keren bro",
6       "user_id": 1,
7       "photo_id": 1,
8       "created_at": "2022-10-21T22:36:04.96213+07:00",
9       "updated_at": "2022-10-21T22:36:04.96213+07:00",
10      "User": {
11        "user_id": 1,
12        "username": "test",
13        "email": "test@gmail.com"
14      },
15      "Photo": {
16        "photo_id": 1
17      }
18    }
19  ]
20 }
```

Update Comment

PUT : <http://localhost:3000/comments /1>

Use Authorization : Bearer Token

PUT

▼

http://localhost:3000/comments/1

Send ▼

Params

Authorization ●

Headers (10)

Body ●

Pre-request Script

Tests

Settings

Cookies

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

Beautify

```

1  {
2    "message": "Sip Update bro!"
3  }

```

Body

Cookies

Headers (3)

Test Results

200 OK 334 ms 264 B

Save Response ▼

Pretty

Raw

Preview

Visualize

JSON ▼

≡

```

1  {
2    "data": {
3      "comment_id": 1,
4      "message": "Sip Update bro!",
5      "user_id": 1,
6      "photo_id": 1,
7      "updated_at": "2022-10-21T22:38:23.2183856+07:00"
8    },
9    "status": 200
10 }

```

Delete Comment

Delete : <http://localhost:3000/comments /1>

Use Authorization : Bearer Token

The screenshot displays the Postman application interface. At the top, the request method is set to 'DELETE' and the URL is 'http://localhost:3000/comments/1'. The 'Authorization' tab is active, showing a 'Bearer...' token type. A warning message is present: 'Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables'. The 'Body' tab is also visible, showing a JSON response: { 'message': 'Your comment has been successfully deleted', 'status': 200 }. The status bar at the bottom indicates a 200 OK response with 357 ms latency and 192 B body size.

SocialmediaController

Post Social Media

POST : <http://localhost:3000/socialmedias>

Use Authorization : Bearer Token

POST <http://localhost:3000/socialmedias> Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 {
2   "name": "Ari Cahyono",
3   "social_media_url": "Ari Cahyono"
4 }
```

Body Cookies Headers (3) Test Results 201 Created 19 ms 287 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "social_media_id": 1,
4     "name": "Ari Cahyono",
5     "social_media_url": "Ari Cahyono",
6     "user_id": 1,
7     "created_at": "2022-10-21T22:39:25.9290949+07:00"
8   },
9   "status": 201
10 }
```

Get Social Media

GET : <http://localhost:3000/socialmedias>

Use Authorization : Bearer Token

GET <http://localhost:3000/socialmedias> Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Type Bearer... Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Body Cookies Headers (3) Test Results 200 OK 22 ms 423 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "social_media_id": 1,
4     "name": "Ari Cahyono",
5     "social_media_url": "Ari Cahyono",
6     "user_id": 1,
7     "created_at": "2022-10-21T22:39:25.9290949+07:00",
8     "updated_at": "2022-10-21T22:39:25.9290949+07:00",
9     "User": {
10      "user_id": 1,
11      "username": "test",
12      "profile_image_url": "BpLnfgDsc2WD8F2qN-1666366037.jpg"
13    }
14   }
15 }
```

Update Social Media

PUT : <http://localhost:3000/socialmedias/1>

Use Authorization : Bearer Token

The image shows a Postman interface for a PUT request. The request is to `http://localhost:3000/socialmedias/1` with a Bearer token. The body is a JSON object with `name` and `social_media_url` fields. The response is a 404 Not Found status with a message: "Comment doesn't exist".

```
PUT http://localhost:3000/socialmedias/1
```

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 {
2   "name": "Ari Cahyono updated2",
3   "social_media_url": "https://warped-shuttle-459668.postman.co/workspace/My-Workspace-df5bdd2a-81a7-4bf5-8ce0-4247d65cf68b/request/10953508-e4194a41-e710-4db5-810d-7338244fa67d"
4 }
```

Body Cookies Headers (3) Test Results 404 Not Found 345 ms 489 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Comment doesn't exist",
3   "status": 400
4 }
5 {
6   "data": {
7     "social_media_id": 1,
8     "name": "Ari Cahyono updated2",
9     "social_media_url": "https://warped-shuttle-459668.postman.co/workspace/My-Workspace-df5bdd2a-81a7-4bf5-8ce0-4247d65cf68b/request/10953508-e4194a41-e710-4db5-810d-7338244fa67d",
10    "user_id": 1,
11    "updated_at": "2022-10-21T22:44:33.6664547+07:00"
12  },
13  "status": 200
14 }
```

Delete Social Media

Delete : <http://localhost:3000/socialmedias/1>

Use Authorization : Bearer Token

The image shows a Postman interface for a DELETE request. The request is to `http://localhost:3000/socialmedias/1` with a Bearer token. The response is a 200 OK status with a message: "Your social media has been successfully deleted".

```
DELETE http://localhost:3000/socialmedias/1
```

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Type Bearer... Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Body Cookies Headers (3) Test Results 200 OK 326 ms 197 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Your social media has been successfully deleted",
3   "status": 200
4 }
```